



Modifications of the Lethality Server for Initial RDEC Federation Integration

Geoffrey C. Sauerborn

ARL-MR-522

DECEMBER 2001

20020204 035

Java® is a registered trademark Sun Microsystems, Inc.

**The findings in this report are not to be construed as an official Department of the Army position
unless so designated by other authorized documents.**

**Citation of manufacturer's or trade names does not constitute an official endorsement or approval of
the use thereof.**

Destroy this report when it is no longer needed. Do not return it to the originator.

Army Research Laboratory
Aberdeen Proving Ground, MD 21005-5066

ARL-MR-522

December 2001

Modifications of the Lethality Server for Initial RDEC Federation Integration

Geoffrey C. Sauerborn
Weapons and Materials Research Directorate

Approved for public release; distribution is unlimited.

Abstract

This report summarizes recent changes in the U.S. Army Research Laboratory distributed interactive simulation lethality communications server (the lethality server) and its integration into the Research, Development, and Engineering Center Federation.

ACKNOWLEDGMENTS

The Research, Development, and Engineering Center (RDEC) Federation is a team effort. Although this report addresses only a single component (the distributed interactive simulation [DIS] lethality server), that component would not have been possible except for the support, contributions, and hard work of many individuals, a few of whom are mentioned here with the author's grateful thanks:

Special thanks to Janet Lacetera and Pat Jones of the U.S. Army Research Laboratory (ARL) and particularly to Paul Oxenberg, Aberdeen Test Center Virtual Proving Ground team leader, for vision, direction, and support in these efforts.

Mr. Richard Sandmeyer of ARL is thanked for his outstanding contributions in coordinating and administering ARL's overall effort with the RDEC Federation and for being a general "unsung hero" regarding modeling and simulation promotion at ARL.

Mark Thomas of ARL is thanked for his coordination in (addition to technical work on the dismounted infantry simulation [DISIM]).

Gary Moss of ARL is thanked for his Java[®] support. The lethality server Java[®] client would have taken many times longer to develop if it were not for his expertise and active willingness to help.

Ken Smith of ARL is recognized for his original design and implementation of the `dis_manager` and most of the DIS manager libraries (ARL-TR-780). Others who have greatly contributed to the manager over the years include Holly Ingham, James Bowen, Kevin Brown, Mark Thomas, and Gary Moss.

Michael Muuss (posthumously), Chuck Kennedy, Jerry Clark, and Doug Kingston of ARL are thanked for their development of the "pkg" library (used by the DIS manager data transport application program interfaces).

Grateful appreciation is expressed for the dedication of Oanh Tran, Gilbert Gonzalez, Pam Nguyen, and many others from the Simulation, Training, and Instrumentation Command for their support during the conference and hosting of the pre-conference integration.

Greg Tackett and Nancy Boucher of the Armament, Munitions, and Chemical Command, Richard Pei of the Communications-Electronics Command, Michael Kelly of the Night Vision Laboratories, and too many more to list are thanked for managing and shouldering much of the unseen administrative navigation required in order to consolidate an effort such as the RDEC Federation.

INTENTIONALLY LEFT BLANK

Contents

1.	Introduction	1
2.	The RDEC Federation	1
2.1	Static Integration Component	1
2.2	Dynamic Interoperation Component	2
3.	The DIS Lethality Server	4
3.1	Disadvantages Addressed	6
4.	Integration	7
5.	Recent Changes in the Lethality Server	9
6.	Java® Client Application	15
7.	Lessons Learned and Possible Future Enhancements	17
7.1	Improving the Lethality Server's Ease of Integration	19
7.2	Need to Explore Alternate Lethality Table Referencing Methods ..	20
8.	"On-the-fly" Verification and Error Checking	21
9.	Summary	22
	References	23
	Appendix	
	A. Photos of Simulation Testing	25
	Distribution List	31
	Report Documentation Page	35
	Figures	
1.	Concept Evaluation in an Operation Scenario	2
2.	Current RDEC Federation Models and Simulations	4
3.	VL Server Design: Client Application's View	8
4.	DIS Monitor Component of the VL Server	9
5.	Standard Damage Entity State Report	10
6.	Entity Heartbeat Update	13
7.	Damage Source Report	13
8.	Simple Java® Client Detonation Report Table	16
9.	Graphical Display of Mobility-Firepower-Catastrophic Probability Distribution for the Selected Impact Event	17
10.	Another, More Detailed View of the Lethality Server's Organization.	18

Tables

1.	Some Advantages of a Lethality Server	5
2.	Some Disadvantages of a Lethality Server	6
3.	Dis_Mon: Entity State Report Key	11
4.	Simulation Environment Information Recently Added to the DIS Monitor	12
5.	Dis_Mon: "Entity Heartbeat Update" Report Key	14
6.	Dis_Mon: "Damage Source Report" Report Key	15
7.	Live Exercise Support Functions	21

MODIFICATIONS OF THE LETHALITY SERVER FOR INITIAL RDEC FEDERATION INTEGRATION

1. Introduction

This report summarizes recent developments in Army modeling and simulation (M&S) objectives and capabilities, especially as they relate to U.S. Army Research Laboratory (ARL) participation in the Research, Development, and Engineering Center (RDEC) Federation project. While this text touches on certain aspects of the RDEC Federation, it is not the author's intent or purpose to discuss or define the RDEC Federation. Instead, focus is on the ARL distributed interactive simulation (DIS) lethality communications server (the lethality server) and its integration into the RDEC Federation.

Although ARL has plans to implement a similar interface in its high fidelity survivability models, the lethality server system currently implements only pre-calculated tabular "look-up" vulnerability results. Factors influencing the implementation of the future higher fidelity interface are not addressed here.

2. The RDEC Federation

The purpose of the RDEC Federation is to support the Army's needs in the areas of design, development, testing, and validating future system concepts and virtual prototypes under the simulation-based acquisition (SBA)/simulation and modeling for acquisition research and training (SMART) process. To accomplish this, a set of applications is required to address specific areas of interest and to simulate them throughout the acquisition cycle. This is the dynamic aspect to the RDEC Federation. There is also a static (non-simulation run time) component.

2.1 Static Integration Component

The static component of the RDEC Federation is the set of items that basically are input to simulations. All participating systems (command, control, communications, computers, and intelligence systems, automotive, armaments, sensors, etc.) have data descriptions that are known before a simulation begins. These descriptions are the performance data (and other system descriptions such as vehicle geometries, terrain locations, scenario descriptions), along with the specifications for file formats, application versions, and other conventions that must be established before run time. This must be accomplished to establish a

The approach is to define the architecture as the interface protocols and objects and to allow compliant applications to join (and leave) the network as required. This approach is used by a number of important Department of Defense (DoD) programs, including the Virtual Proving Ground (VPG) [1], Test Enabling National Architecture (TENA) [2], and others. VPG and TENA are named explicitly because efforts to ensure a level of compatibility between these environments and the RDEC Federation are ongoing as it develops.

The philosophy of defining the interfaces and allowing simulations to join as needed augments the distributed simulation phenomenon that has been developed over the past couple decades. It is a natural progression from simulation network (SIMNET) to the establishment of the DIS standard, and culminating (so far) with the DoD high level architecture (HLA) standard [3,4,5]. In fact (concerning run time protocols), the current RDEC Federation version (as it was implemented during the SMART 2001 exercises) is a hybrid of DIS and HLA. The "Federation" portion of the RDEC Federation name is a direct application of the HLA Federation [6].

From a network architecture viewpoint, the important job is to standardize data objects and other specifics that define the interfaces. However, this aspect of the system may be of little importance to a VPG or RDEC Federation customer. The customer's interest chiefly concerns "what can the system do for me?" This translates to "what can the overall system simulate, and will this meet my needs?" Therefore, the customer is interested in the applications that are available. Current RDEC Federation applications are a diverse set of high fidelity virtual and constructive models and utilities operating in multiple force-on-force simulation environment. Depicted in Figure 2 is the RDEC Federation M&S environment capability used to exercise notional future combat system concepts demonstrated during the SMART 2001 conference 16-19 April 2001, Orlando, Florida.

Simulations and applications are depicted (without further explanation) as being attached to the DIS and/or HLA network¹. With a subset of these applications, various exercises involving future system concepts were used to demonstrate the RDEC Federation concept and its current capabilities during the conference.

¹Responsible U.S. Army agencies are listed below:

AMCOM – Aviation and Missile Command

CECOM – Communications Electronics Command

STRICOM - Simulation, Training & Instrumentation Command

TARDEC - Tank Automotive Research, Development, and Engineering Center

ARDEC – Armament Research, Development and Engineering Center

ARL – Army Research Laboratory.

(Note: ARL's ground systems test bed component is a VPG project being developed jointly with the Developmental Test Command [DTC].)

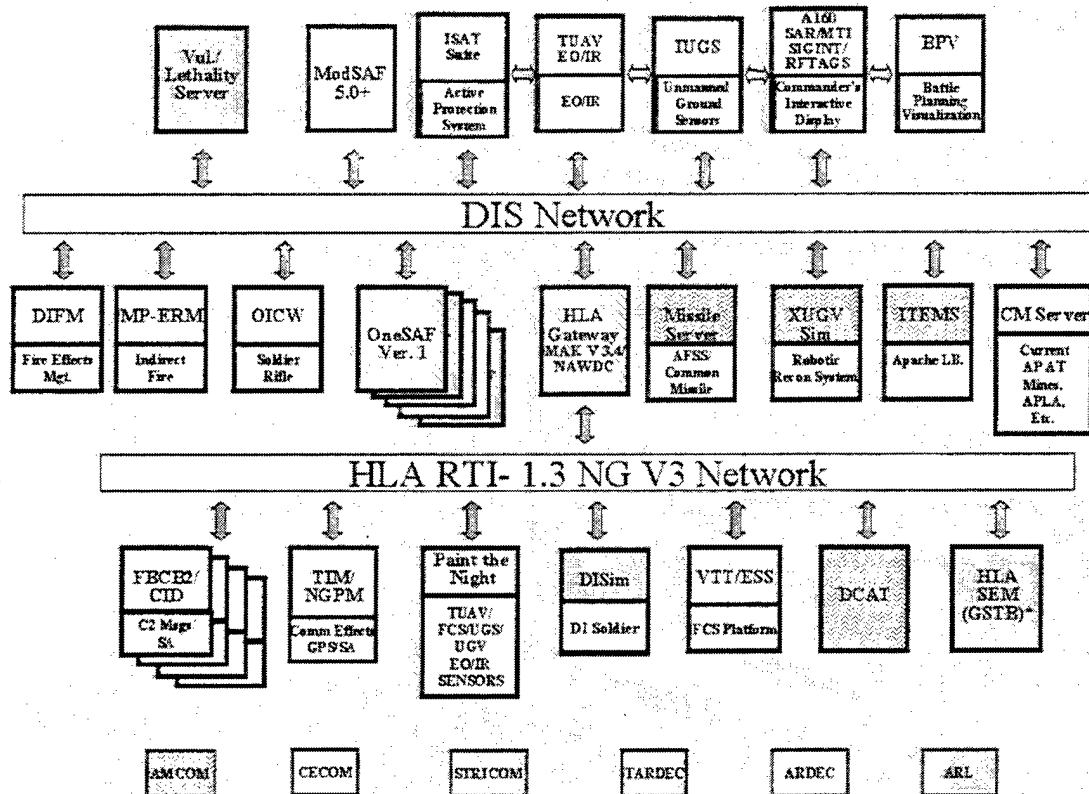


Figure 2. Current RDEC Federation Models and Simulations.

Figure 2 depicts the HLA/DIS run time components of the RDEC Federation at the time of the SMART conference. As explained in section 2.1 the Federation also has a static (pre-run time) integration component not displayed in Figure 2. This text is not intended to be a description of either component especially as they are still in development and their character and composition are expected to evolve with time. They have been presented here merely as a background and context from which to address the DIS Lethality Server.

3. The DIS Lethality Server

The component application that we are focusing on is the lethality server. The server is designed for the DIS environment. As such, the server connects on the DIS side of the RDEC Federation. The server is a combination of application program interface (API) libraries and utility programs that make it possible to allow multiple applications to access a single lethality data source. The server delivers pre-calculated lethality outcomes resulting from any combination of relevant parameters in near real time (on the order of 1/100th of a second), making it suitable for most real time and human-in-the-loop applications [7]. In this way, the server can be used to uncouple the damage calculation component from simulations in the distributed environment. However, since damaged results are normally pre-calculated (and stored in "look-up" tables) inside each

combat simulation, one may ask what advantage is there for a lethality server. The lethality server offers several advantages by de-coupling the damage component. These advantages (and some disadvantages) are outlined in Tables 1 and 2.

Table 1. Some Advantages of a Lethality Server

Lethality Server Advantage Explained	
A-1	The server would have the potential to eliminate DIS interoperability variances in lethality outcomes (remove "unfair" weapons effects play), since all DIS simulations will resolve lethality effects through a single, repeatable means.
A-2	It would allow increased ease of verification, validation, and accreditation for battle simulations exercises as a result of having a standard (and centralized) set of unclassified lethality calculations.
A-3	It could decrease DIS simulation development time by providing a complete, computer platform-generic, vulnerability/lethality handling mechanism. That is, because the lethality issue has been decoupled from the rest of the simulation, the lethality handling mechanisms may be "stubbed out," allowing more time to be devoted to the rest of the simulation development process.
A-4	Higher fidelity lethality results can be implemented with little modification of the way a simulation receives that information. The design of the current table look-up server contains a means whereby other results (that are not precalculated) could be implemented. That is, an application could use the same server interface to receive higher fidelity results if desired (calculated, for example, from a remote process). However, there is currently no implementation beyond precalculated look-up tables.
A-5	Using different damage descriptions (beyond just M,F,K) can be implemented with little modification of the way a simulation receives that information. In the current server design, there is a means to add different damage descriptions (such as less-than-lethal) mechanisms or completely different ways of dividing the lethality "space" [8]. The user manual walks through the steps taken to add a new lethality description [9].

In most combat simulations, lethality and vulnerability have to be implemented somewhere. It does not matter whether it is a first principle calculation or look-up table (based on first principles); either way, erroneous parameters, algorithms, or data sources could be misapplied.

While a centralized VL server has the advantage of “off-loading” the task, it can also off-load the responsibility and create the mistaken assumption that “somebody else” is going to take care of that aspect the process. Ensuring that the data are there and are being sought or calculated correctly must be part of the VL server’s implementation and management.

Table 2. Some Disadvantages of a Lethality Server

Disadvantage of a Lethality Server	
D-1	Client applications (and their operators and sponsors) will have to “trust” the results returned by the server. Errors may still occur in the data population or return of vulnerability results; since the server may not be under a developer’s direct configuration, the error can very well be more difficult to trace (or even detect).
D-2	Response to a VL query (and therefore simulation execution time) might be longer than without a server. Adding another software layer will almost never increase overall application speed, especially when network communications are a part of that added layer.
D-3	There is a danger of erroneous data because of a general lack of attention.

3.1 Disadvantages Addressed

3.1.1 D-1 Addressed

Certain enhancements can be added to the server to help avoid disadvantage D-1. Low-level “debugging” APIs are already implemented, which provide vulnerability parameter information (namely, `vlp_print_all_params()` [9, Appendix B, `VLParam(3)`]). In order for a remote client to “check the results,” this type of information needs to be distributed (provided by a client query as opposed to an API call). Other information should also be accessible. For example, if the result is derived from a look-up table, then the exact source of the table (location and file or record name) along with the algorithm used to obtain the results from that table should be available to the client.

While these enhancements are necessary for a client to scrutinize a lethality server’s vulnerability results, they are not sufficient. In the author’s opinion, sufficiency would only be approached once these enhancements are “packaged” in a separate tool complete with an intuitive, easily operated, graphical user interface (GUI).

3.1.2 D-2 Addressed

Accessing vulnerability data across a network will most certainly be slower than accessing locally (and inside a simulation). Applications that are highly time dependent or tightly coupled with other components that are faster than real time will need to seriously consider the time limitations of a distributed server. However, applications that are so tightly coupled may never be part of a distributed environment such as DIS or HLA. If they are, then the timing limitations may still be within their tolerance (the exact limitations are system dependent).

The server provides a means to be operated without using the network communications component. This is accomplished by calling VL (table look-up) API functions and linking them directly into one's application [7]. Of course, doing so removes all the advantages listed in Table 1. Still, this might be a consideration if the server has certain data look-up algorithms or its database is populated with certain data that an application requires. These may be applied without added software development or data maintenance overhead.

3.1.3 D-3 Addressed

This is largely a systemic issue as opposed to a technical question. Errors can enter into the process whether the VL calculation is conducted locally or on some remote server. Using a VL server can be more efficient because the VL configuration and management are done once per target-threat and do not have to be repeated among all participants.

However, it does not grant participants a license to forget about results. Some automated tools that allow quick "sanity checks" of VL results have already been suggested in the response to D-1.

4. Integration

This section addresses how the lethality server was integrated into the RDEC Federation exercises conducted during the SMART conference.

During normal operation, the DIS server is designed to work in a client/server mode as depicted in Figure 3. The intent is for simulations (or utilities) to act as "clients". These clients query for the results of a particular detonation or munitions impact. The server monitors the battlefield environment. It therefore knows who shot whom with what and can return the results of the detonation to the querying client. This assumes that the appropriate results have been loaded into the server's database ahead of time.

While the advantages listed in Table 1 are attractive, none of the other RDEC Federation applications were prepared to use the server during the SMART 2001 demonstration. Although VL client implementation is straightforward and basically involves only two fundamental API calls (`vls_send()` and `vls_receive()` to send a query and receive the answer, respectively), it does involve some programming and testing. None of the participants were able or prepared to implement this level of integration in time for SMART.

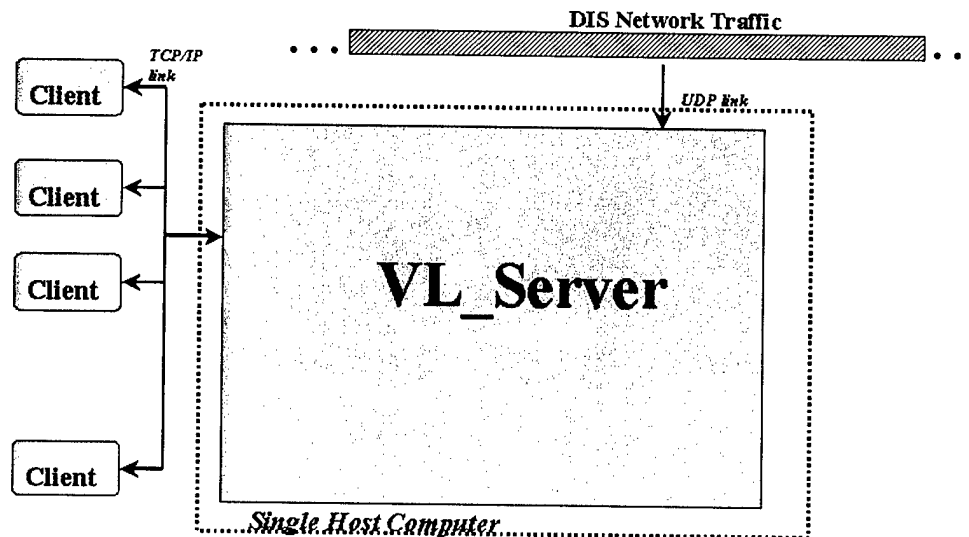


Figure 3. VL Server Design: Client Application's View.

Furthermore, during integration tests in preparation for the SMART exercise, it was decided to physically separate the two networks shown in Figure 2 (DIS and HLA networks). The reasons for this were because some HLA applications were not able to function properly as a result of the heavy amount of DIS traffic on the same network. The result was that HLA and DIS applications were unable to communicate with each other². However, since lethality server clients operate via other means (socket connections over a transmission control protocol/internet protocol link provided by the server's API), no HLA application could communicate with the server since it was on the DIS sub-network. This had no impact on the server's role during the SMART tests since the server was basically in a monitoring role and its only client application was the Java[®] Client that resided on the DIS network (see Section 6).

²Except via the HLA/DIS bridge seen as the "HLA Gateway" in Figure 2.

5. Recent Changes in the Lethality Server

In its monitoring role, the server comes equipped with an application that monitors the DIS environment and pays special attention to lethality-dependent information (who shot whom with what and the conditions at the time). This application is called the DIS monitor. The DIS monitor's place within the overall lethality server architecture is depicted in Figure 4.

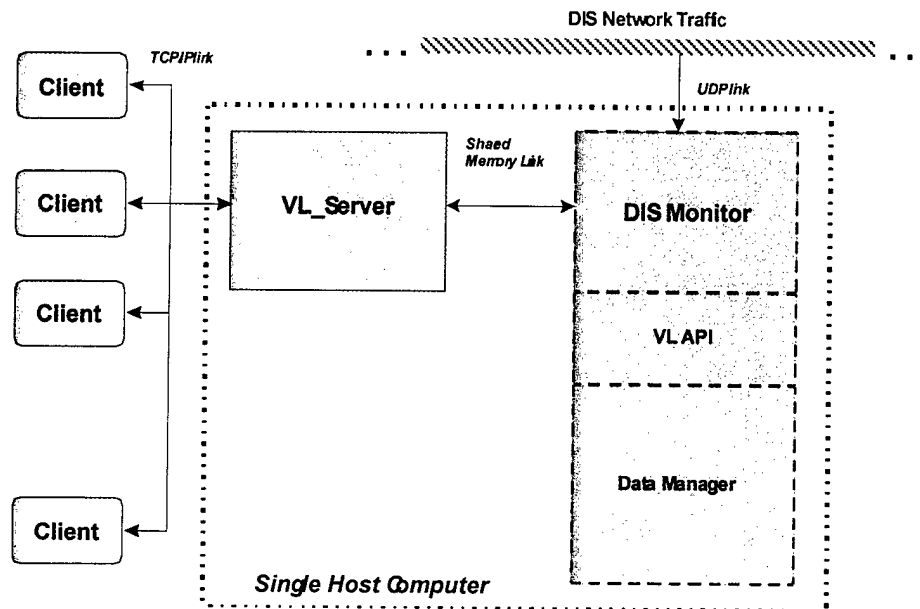


Figure 4. DIS Monitor Component of the VL Server.

Because the DIS monitor “knows about who shot whom with what and when,” it can use the VL API to service lethality look-up table lethality results and supply them to the VL_Server. This is how the VL_Server supplies results to clients. The VL_Server does not actually do any calculating; it merely passes queries (to some destination; in this case, the DIS monitor) and returns the results.

Another capability the DIS monitor has is its ability to display some of its real-time “knowledge” of the battlefield state through interactive screen reports. Figure 5 shows a screen capture of one of these reports (the Standard Entity Damage State report). Each row of this report displays the state of a single entity. Since the DIS monitor is only monitoring the battlefield and is not affecting it, the state information displayed shows the damage state as reported by each entity. This is not necessarily the damage as calculated by the VL server, since the entity may not have queried the server but calculated its own damage in some manner. The Damage State report columns are explained in Table 3.

To support the SMART exercises, additional data items were added to the DIS monitor's items of interest. These items tracked how often and when the last update occurred from each entity. Therefore, these data could be applied as a basic simulation environment monitor. The newly tracked data shown are in Table 4. These data were then used to create a new report the "Entity Heartbeat Update," as seen in Figure 6. Table 5 defines the data in the "Entity Heartbeat Update" report.

dis_mon

tracking 243 Entities in Exercise 1

Thu Sep 13 10:31:54 EDT 2001

--Entity----- --KILL--- --Damage----- --Smoke---- Times
Frc ID Type Mobil FireP Slight Modrt Dstryd Plm Eng PlmEng Hit

1 1167 "MRAS", 0 0 0 0 0 0 0 0 0 0
1 1086 "RAVE_Mortar", 0 0 0 0 0 0 0 0 0 0
1 1180 unknown 0 0 0 0 0 0 0 0 0 0
1 1019 "IUGS_IR", 0 0 0 0 0 0 0 0 0 0
1 33 "DI_Oobs_Post", 0 0 0 0 0 0 0 0 0 0
1 1001 "M1 Abrams", 0 0 0 0 0 0 0 0 0 0
1 1016 "IUGS_IR", 0 0 0 0 0 0 0 0 0 0
1 1031 "A160", 0 0 0 0 0 0 0 0 0 0
1 1012 "IUGS_SA", 0 0 0 0 0 0 0 0 0 0
1 1108 "RAVE TT MCU", 0 0 0 0 0 0 0 0 0 0
1 1025 "IUGS_IR", 0 0 0 0 0 0 0 0 0 0
1 2 "Advanced Field 0 0 0 0 0 0 0 0 0 0
1 16384 "DI_M16A2", 0 0 0 0 0 0 0 0 0 0
1 6 "McDonnell-Doug 0 0 0 0 0 0 0 0 0 0
1 1105 "RAVE_BLOS", 0 0 0 0 0 0 0 0 0 0
1 1110 "RAVE_Recon", 0 0 0 0 0 0 0 0 0 0
2 1267 "BMP-2", 1 1 1 1 1 1 1 0 1 0
2 1240 "BMP-2", 1 0 1 1 1 1 1 0 1 0
2 1150 "T-80 MBT", 1 1 0 0 0 0 0 0 0 0
2 1001 "2S6 Quad 30-mm 0 0 0 0 0 0 0 0 0 0
2 1151 "T-80 MBT", 1 0 1 1 1 1 1 0 1 0
2 1258 "BMP-2", 0 0 0 0 0 0 0 0 0 0
2 1265 "BMP-2", 1 0 1 1 1 1 1 0 1 0
2 1245 "BMP-2", 0 0 0 0 0 0 0 0 0 0
2 1152 "T-80 MBT", 1 0 1 1 1 1 1 0 1 0
2 1302 "BMP-2", 0 0 0 0 0 0 0 0 0 0
2 10 "T-80 MBT", 0 0 1 1 1 1 1 0 1 0
2 1268 "BMP-2", 1 1 1 1 1 1 1 0 1 0
2 1178 "T-80 MBT", 1 1 1 1 1 1 1 0 1 0
2 1276 "BMP-2", 1 0 1 1 1 1 1 0 1 0
2 1249 "BMP-2", 1 1 1 1 1 1 1 0 1 0

FRIENDLY FOES
(Force ID 1) (Force ID 2)
(blue) (red)

KKilled 0 11
MKilled 0 1
FKilled 0 1
MFKilled 0 2

rollup.out

Figure 5. Standard Damage Entity State Report.

Since the Standard Damage Status Report (see Figure 5) does not track how (or what caused) the damage, a supplemental report was added: the Damage Source Report (see Figure 7). This report did not require additional internal data fields to

be added to the DIS Monitor—just the report page. The fields for this report are explained in Table 6.

Table 3. Dis_Mon: Entity State Report Key (from Figure 5)

Column	Meaning
Entity	
Frc	This is the "Force" identification of an entity (whose "side" an entity is on during a battle). Data in this field come from the "Force ID Field" of the "Entity State PDU" (protocol data unit). Valid Force IDs are 0 = Other 1 = Friendly 2 = Foe 3 = Neutral
ID	This is the Entity ID Field portion of the PDU's Entity Identifier Record (a three-integer record). These three integers represent the simulated entity's SITE, HOST, and APPLICATION. Their combination uniquely identifies an entity [4, Section 5.3.14.2 "Entity Identifier"]. The "ID" integer in this column is actually only last of the three 16-bit unsigned integers ("APPLICATION").
Type	This column reports the name of the entity type. The entity type is a numeric value defined in the Entity Type Record (of the Entity State PDU). The name seen in this column is the text name associated with that numeric entity type ID. The text name comes from the VL Data Manager initialization file's "DIS_ENTITIES_FILE" record [9, vls_db_init(5)].
KILL	
Mobil	Bool: Set to true (1) if entity is mobility killed
FireP	Bool: Set to true (1) if entity is fire power killed
Damage	Slight Bool: Set to true (1) if entity is slightly damaged
Modrt	Bool: Set to true (1) if entity is moderately damaged.
Dstryd	Bool: Set to true (1) if entity is destroyed.
Smoke	
Plm	Bool: Smoke plume is rising from the entity.
Eng	Bool: Entity is emitting engine smoke.
PlmEng	Bool: Entity is emitting engine smoke and smoke plume is rising from the entity.
Times Hit	This field displays the number of times that dis_mon saw the entity "hit" by a munition. This is a derived number and does not appear in the Entity State PDU ³ .

³Note, Figure 6 shows zero hits under the "Times Hit" tally because (in this instance) dis_mon started monitoring the simulation after the hits occurred. Thus, the actual damage-causing detonations were never observed (and thus the number of hits were untallied and are unknown). What is known is the published damage state, as updated by the entity and reflected in "Entity

Table 4. Simulation Environment Information Recently
Added to the DIS Monitor

Data Item	Description	Purpose
T_Last	Time since last entity state update was seen. This is monitored for each entity on the virtual battlefield.	Used to monitor for client inactivity (and possible time-out). <u>Displayed in:</u> "Time Since Last PDU" column of "Live Exercise Status Updates" report. – Figure 6
Entity State PDU Count	The total number of state updates broadcast by an entity. This is monitored for each entity on the virtual battlefield.	May identify sources of heaviest network activity. Certain entities by virtue of their mission (e.g., a moving versus a stationary entity) may produce orders of magnitudes more network data than others. <u>Displayed in:</u> "COUNT ES PDUs" report column (see Figure 6).
EntityID	Displays three integers representing the controlling entity's Site (location), Host (computer), and Application (program or simulation).	It is often useful to identify the originating computer system or application from simulation entities. In particular, when those entities are displaying "misbehavior" or are not intended to be a participant in the current exercise ⁴ . <u>Displayed in:</u> "SITE HOST APP" report column (see Figure 6).

⁴It is the responsibility of the DIS application to fill these fields before updates are published. However, a problem arises in identifying the source of these PDU updates since by definition, the application may be "rogue" (running out of control) and therefore may not be filling these fields with the proper data. (This actually happened during the pre-SMART conference integration tests at STRICOM.) An enhanced identifier scheme would also track the originated IP address from the user datagram protocol header field. However, this would require some modifications in order to pull these data from the Muss, Kennedy, Clark, and Kingston (the pkglib) libraries; time did not allow this.

315.000

tracking 243 Entities in Exercise 1

Thu Sep 13 10:29:45 EDT 2001

--Entity--

Frc	ID	Type	Marking	DIS Enumeration	---TIME-(sec)- Since Last PDJ	-COUNT- ES PDUs	SITE	HOST	APP
1	1167	"MRAS"	100B14"	(1,1,225,4,8,1,0)	31,293407	82	10	2	1167
1	1090	"MRAS"	100B21"	(1,1,225,4,8,1,0)	31,963518	71	10	2	1090
1	1197	"MRAS"	100B12"	(1,1,225,4,8,1,0)	32,434893	77	10	2	1197
1	1208	"MRAS"	100A11"	(1,1,225,4,8,1,0)	31,175603	76	10	2	1208
1	3034	"AGM114L"	ITEMS"	(2,2,225,1,3,5,2)	88,273417	21	10	61	3034
1	3037	"AGM114L"	ITEMS"	(2,2,225,1,3,5,1)	40,563576	80	10	61	3037
1	1193	"MRAS"	100A12"	(1,1,225,4,8,1,0)	31,923411	64	10	2	1193
1	1203	"MRAS"	100A14"	(1,1,225,4,8,1,0)	31,903468	74	10	2	1203
1	1192	"MRAS"	100B11"	(1,1,225,4,8,1,0)	31,883564	85	10	2	1192
1	1188	"MRAS"	100A13"	(1,1,225,4,8,1,0)	31,113591	75	10	2	1188
1	1182	"MRAS"	100B13"	(1,1,225,4,8,1,0)	31,373299	79	10	2	1182
1	1129	unknown	100D14"	(1,1,225,6,1,30,3)	33,463511	2	10	2	1129
1	1175	unknown	100B41"	(3,1,225,1,81,1,0)	75,623369	5	10	2	1175
1	3035	"AGM114L"	ITEMS"	(2,2,225,1,3,5,2)	73,663513	15	10	61	3035
1	1186	"DI Rifle"	100B32"	(3,1,225,1,32,1,1)	34,633363	12	10	2	1186
1	1229	"DI Rifle"	100A42"	(3,1,225,1,32,1,1)	46,3540	8	10	2	1229
1	1187	unknown	100B31"	(3,1,225,1,81,1,0)	49,883611	3	10	2	1187
1	1230	unknown	100A41"	(3,1,225,1,81,1,0)	45,473564	14	10	2	1230
1	1174	"DI Rifle"	100B42"	(3,1,225,1,32,1,1)	33,613450	24	10	2	1174
1	1179	"DI Rifle"	100A32"	(3,1,225,1,32,1,1)	34,213533	14	10	2	1179
1	1086	"RAVE_Mortar"	100A61"	(1,1,225,3,21,7,0)	107,133084	2	10	2	1086
1	1180	unknown	100A31"	(3,1,225,1,81,1,0)	54,173388	3	10	2	1180
1	1016	"IUGS_IR"	100S12"	(5,1,0,1,9,3,0)	65,483543	3	10	98	1016
1	1019	"IUGS_IR"	IUIS2"	(5,1,0,1,9,3,0)	66,323370	3	10	98	1019
1	1106	"RAVE_BLOS"	100D41"	(1,1,225,4,21,4,0)	68,833553	3	10	2	1106
1	33	"DI Obs Post"	COLT39"	(3,1,225,2,0,0,0)	65,183472	2	28	135	33
1	1001	"M1 Abrams"	firebird"	(1,1,225,1,1,1,0)	31,413236	9	10	91	1001
1	1188	"RAVE TT_MCU"	100B21"	(1,1,225,4,8,3,2)	38,453415	8	10	2	1188
1	1116	"RAVE_BLOS"	100G12"	(1,1,225,4,21,4,0)	60,803474	3	10	2	1116
1	1031	"A160"	A160"	(1,2,225,50,20,0,0)	42,303434	3	10	98	1031
1	1113	"RAVE_BLOS"	100H12"	(1,1,225,4,21,4,0)	60,543513	3	10	2	1113
1	2	Advanced Field	2/1/C/2 /4"	(1,1,225,4,9,0,0)	59,863123	2	28	135	2
1	16384	"DI M16A2"	ARL_D101"	(3,1,225,1,32,1,0)	42,203378	4	10	44	16384
1	6	McDonnell-Doug	AH64Db"	(1,2,225,20,1,4,0)	54,853538	2	10	61	6
1	1189	"RAVE TT_MCU"	100A22"	(1,1,225,4,8,3,2)	34,313503	4	10	2	1189
1	1105	"RAVE_BLOS"	100B51"	(1,1,225,4,21,4,0)	102,643517	2	10	2	1105
1	1107	"RAVE TT_MCU"	100G32"	(1,1,225,4,8,3,2)	57,103442	3	10	2	1107
1	3	Advanced Field	3/1/C/2 /4"	(1,1,225,4,9,0,0)	57,543148	2	28	135	3
1	1089	"RAVE_Mortar"	100B62"	(1,1,225,3,21,7,0)	58,313281	2	10	2	1089
1	1114	"RAVE_BLOS"	100H13"	(1,1,225,4,21,4,0)	50,663526	2	10	2	1114

315.000

Figure 6. Entity Heartbeat Update (shows the time since the latest entity update was received and origin of the update).

315.000

/ Listening...

PDUs seen: 2994

tracking 239 Entities in Exercise 1

-Event-	-ID-	-Firing_Entity_Type-	-Target_Type-	-Munition_Type-	Type of - -Detonation-
10	2	220	{ (1,1,222,2,2,1,0) }	{ (2,8,222,2,2,2,2) }	3 ("Ground Imp")
10	2	221	{ (1,1,222,2,2,1,0) }	{ (2,8,222,2,2,2,2) }	3 ("Ground Imp")
10	2	222	{ (1,1,222,2,2,1,0) }	{ (2,8,222,2,2,2,2) }	3 ("Ground Imp")
10	2	223	{ (1,1,222,2,2,1,0) }	{ (2,8,222,2,2,2,2) }	3 ("Ground Imp")
10	2	224	{ (1,1,222,2,2,1,0) }	{ (2,8,222,2,2,2,2) }	3 ("Ground Imp")
10	2	225	{ (1,1,222,2,2,1,0) }	{ (2,8,222,2,2,2,2) }	3 ("Ground Imp")
10	2	226	{ (1,1,222,2,2,1,0) }	{ (2,8,222,2,2,2,2) }	3 ("Ground Imp")
10	2	227	{ (1,1,222,2,2,1,0) }	{ (2,8,222,2,2,2,2) }	3 ("Ground Imp")
10	2	228	{ (1,1,222,2,2,1,0) }	{ (2,8,222,2,2,2,2) }	3 ("Ground Imp")
10	2	229	{ (1,1,222,2,2,1,0) }	{ (2,8,222,2,2,2,2) }	3 ("Ground Imp")
10	2	230	{ (1,1,222,2,2,1,0) }	{ (2,8,222,2,2,2,2) }	3 ("Ground Imp")
10	2	231	{ (1,1,222,2,2,1,0) }	{ (2,8,222,2,2,2,2) }	0 ("Other")
10	2	232	{ (1,1,222,2,2,1,0) }	{ (2,8,222,2,2,2,2) }	0 ("Other")
10	61	68	{ (1,2,225,20,1,4,0) }	{ (1,1,222,2,2,1,0) }	0 ("Other")
10	61	76	{ (1,2,225,20,1,4,0) }	{ (1,1,222,2,2,1,0) }	1 ("Entity Imp")
10	61	76	{ (1,2,225,20,1,4,0) }	{ (2,2,225,1,3,5,2) }	0 ("Other")

315.000

Figure 7. Damage Source Report (who shot whom with what).

Table 5. Dis_Mon: "Entity Heartbeat Update" Report Key (from Figure 6)

Column	Meaning
Entity	
Frc	This is the "Force" identification of an entity (whose "side" an entity is on during a battle). Data in this field come from the "Force ID Field" of the "Entity State PDU." Valid Force IDs are 0 = Other 1 = Friendly 2 = Foe 3 = Neutral
ID	This is the Entity ID Field portion of the PDU's Entity Identifier Record. ID it is actually only the last of the three 16-bit unsigned integers (SITE, HOST, APP) that identify an entity instance in a DIS exercise. See (SITE, HOST, APP) column in this table [4, Section 5.3.14.2, "Entity Identifier"].
Type	This column reports the name of the entity type. The entity type is a numeric value defined in the Entity Type Record (of the Entity State PDU). The name seen in this column is the text name associated with that numeric entity type ID. The text name comes from the VL Data Manager initialization file's "DIS_ENTITIES_FILE" record [9, vls_db_init(5)].
Marking	Sometimes known as the "bumper number" because it is often informally used to denote the unit designation on vehicles (such as tanks). In terms of the DIS standard, it is the "Entity Marking Field" as published by the issuing application [4, Section 5.3.15 "Entity Marking Record"].
DIS Enumeration	These seven numbers represent the entity type record. The digits represent the subfields "kind," "domain," "country," "category," "subcategory," "specific," and "extra" as defined in the DIS standard [4, Section 5.3.16 "Entity Type Record"]. Their exact interpretation depends on the enumeration standard used [5] with modifications conventional to the current exercise.
TIME Since Last PDU	This field shows the time (in seconds) since the DIS monitor last detected a state update (an Entity State PDU) from the entity.
COUNT ES PDUs	The number in this field represents the total updates (Entity State PDUs) detected for the given entity. Certain entities may issue updates at a greater rate, depending on their actions (for instance, if they are moving [many state changes] versus if they are stationary).
SITE	Together, these three columns represent and identify an entity's unique instance in a DIS exercise. Each entity on the virtual battlefield has a unique "Entity Identifier" record.
HOST	SITE usually denotes a physical location or facility; HOST identifies the host computer system; and APP usually identifies an application that is simulating the entity.
APP	[4, Section 5.3.14.2 "Entity Identifier" record]

Table 6. Dis_Mon: "Damage Source Report" Report Key (from Figure 7)

Column	Meaning
Event ID	Together, the three integers in this column represent and uniquely identify the detonation event. Each time a "fire" or "detonation" event occurs on the virtual battlefield, an Event identifier is issued to identify that event. Sometimes (as is the case with some explosions such as a demolition charge) only a detonation event is issued. Other times, there is a "fire" event associated with a detonation event (such as an artillery munition launched [the "fire"] and its impact [the "detonation"]). When fire and detonation events are related, they have the same "Event ID" [4, Section 5.3.18 "Event Identifier" record].
Firing Entity Type	These seven numbers represent the entity type record that identifies the munition that has detonated (or impacted). The digits represent the subfields "kind," "domain," "country," "category," "subcategory," "specific," and "extra" as defined in the DIS standard [4, Section 5.3.16 "Entity Type Record"]. Their exact interpretation depends on the enumeration standard used [5] with modifications conventional to the current exercise. For instance (1,1,222,2,2,1,0) represents a Russian "BMP-2".
Target Type	This field is the "Entity Type" enumeration for the type of entity that was targeted by the shooting entity. If the firing entity did not specify a target, then "<none>" appears.
Munition Type	The "Entity Type" enumeration of the munition used is entered here.
Type of Detonation	This field specifies an 8-bit unsigned integer. This number is the DIS enumeration describing what kind of detonation occurred. The enumeration as filled by the detonation issuing simulation and a text interpretation of its value is shown. For instance, "3" represents a "Ground Impact" [4, Section 5.4.4.2 , Detonation PDU (10) Detonation Result].

6. Java® Client Application

A simple Java® client was added to the server for the SMART conference. This client application has access to the usual set of queries available to other VL server clients. It also allows access to a versatile GUI API (the Java® graphic environment objects) and allows a platform-independent means to present information that is better explained graphically. For instance, Figure 8 shows that the Java® client employs the "JTable" object from the Java® Swing Toolkit to display a variation of the "Damage Source Report" (see Figure 7).

the battlefield (the DIS monitor). Figure 4 is further detailed in Figure 10 to display the DIS manager's placement in the server's architecture.

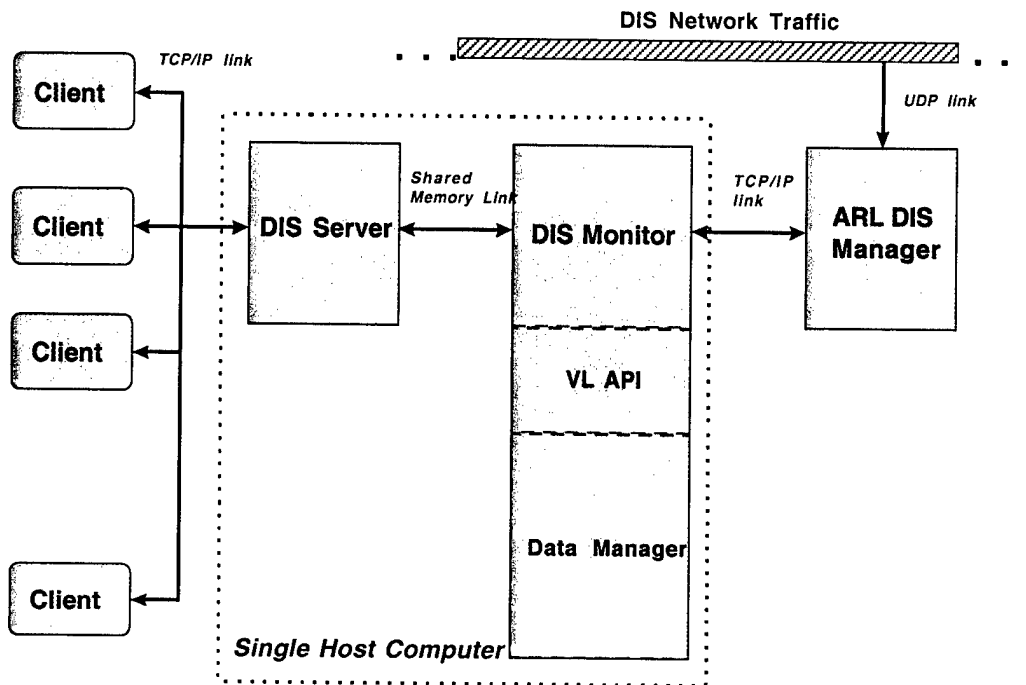


Figure 10. Another, More Detailed View of the Lethality Server's Organization.

Only certain data packets are of interest to the lethality server (these packets represented state changes, fire and detonation events). Other packets were logged but filtered and not passed to the DIS monitor. However, those packets that were passed represented the vast majority of data packets.

Unfortunately, data flow (to clients) fell on the other end of the spectrum in terms of stress testing. This is because none of the other RDEC Federation participating applications and simulations were prepared to use the lethality server in time for the SMART Conference exercises. Only two clients were attached at any one time (and these were merely utility clients that come as part of the lethality server suite: a simple text-based interactive client and the Java® client).

In summary, both the DIS monitor and DIS manager sustained performance with excellence in processing a heavy data inflow. However outflow data volume (a flood of queries) was not sufficiently tested.

Quick analysis was at times cumbersome. For the existing tools, improved interfaces need to be tailored to better access the available information. The server was able to use its simulation monitoring capabilities, but their usefulness

could have been improved with supporting search tools and GUIs. Several times during integration tests, the need arose to determine the state of a particular vehicle or entity or to replay a portion of the exercise for closer examination. The data logging by the DIS manager captures DIS data packets (called PDUs) into a single sequential large binary file. With a playback utility, these PDUs may be replayed at various speeds. However, there is no efficient interface that allows the PDUs to be searched and browsed.

The DIS monitor logs items of interest (fire events, detonations, the munitions used, intended targets, etc.) into its own separate flat sequential text file. However, these data are of select material, which is good as long as all questions are contained within that material. However, all manner of questions cannot be anticipated ahead of time. For example, one particular question arose, "Why didn't *this* munition destroy *that* vehicle?" It became cumbersome in the heat of the moment to manually examine this text log to resolve the issue⁶. While sufficient for post-process review, this flat file of selected material was an inefficient means to quickly search for events related to various entities.

While all information is being captured, it should be better entered in a database (or at least be exportable in extensible markup language to be read by a dedicated database management system).

In summary, the data from all events, including special events (such as detonations), should be logged into a tool more suited for analysis (such as a real database). The two logs files (from the DIS monitor and DIS manager) should be more tightly correlated or even be one and the same, thus allowing all manner of questions to be resolved in a timely manner. The DIS manager's mission playback capabilities could be enhanced by the addition of bookmark, search, and browse capabilities. Currently, the DIS manager's playback can only "play" (or play in a "fast forward" mode). Playback could be enhanced to include the other standard "VCR-like" features: rewind, reverse, and pause. This should further be made useful by wrapping all these features into a familiar looking intuitive and user friendly GUI.

7.1 Improving the Lethality Server's Ease of Integration (into combat models)

Currently, client applications connect to the DIS look-up table lethality server via the server-supplied library of APIs. A small but necessary set of APIs is required in order to query the server: `vls_open()`, `vls_close()`, `vls_send()`, and `vls_read()`. Client applications send text queries to the server via the `vls_send()` API and

⁶Eventually, the issued was resolved. It was discovered that the vehicle never left its line of departure and thus stayed out of "harm's way". Therefore, while many other systems were damaged, this one vehicle remained unscathed. Part of the problem stemmed from determining which entity was "that vehicle"; another was searching for the lethality server results for detonations against "that vehicle," of which, there were none.

retrieve the results by the `vls_read()` function. By integrating the functionality of these APIs into an HLA simulation object model, client applications will no longer have to link the server's APIs directly into their applications. This would lower "integration risks" from the client's viewpoint. Furthermore, it is anticipated that future expansion of RDEC Federation applications (and new applications) will tend toward HLA instead of the DIS standard.

In summary, the lethality server's functionality should migrate to HLA. Because the RDEC reference federation object model (FOM) is currently in flux, that migration should be executed in as flexible a manner as is practical. An HLA "middleware" approach could aid here, provided that it also has a flexible FOM design⁷.

7.2 Need to Explore Alternate Lethality Table Referencing Methods

A shortcut method may be necessary to refer to lethality look-up tables. Currently, the lethality server is initialized with a look-up table for *each* possible combination of munition and target. However, this leads to table references that grow at a rapid rate ($N \times M$). For instance, the Institute of Electrical and Electronics Engineers DIS enumeration standard supported by the server contains enumerations for more than 5,000 separate entities and 1,000 munitions. This produces more than 5 million table references.

The size is only one aspect to consider. This is an internal implementation consideration, but reducing memory requirements could enhance performance and portability. Currently, look-up table references are stored in internal memory. If each table reference can be restricted to 100 characters, this would still require about 500 megabytes of random access memory to store⁸. Ways of reducing this internal memory requirement could be explored. In practice, this has not been a problem since all entity and munition types have never appeared in a single exercise. The largest SMART conference scenarios had hundreds of entities. However, this required only ~50 entity and 25 munition types. Still, this produced more than 1,000 references, which proved at times too complicated to manage within a flat data file.

Managing look-up table references could be vastly improved by adding a tool. This GUI control station for the server would oversee the server's operator configuration and management. It could check for errors and replications during the lethality data population phase. Default settings (for the case when no VL data are available) could be implemented.

⁷FOM flexibility (or neutrality) refers to the degree to which an application can use any number of FOMs. A FOM neutral application is not "hard coded" to any particular FOM and thus can be readily adapted when FOM changes are made.

⁸One hundred characters times 5 million table references or about 500 million characters.

8. "On-the-fly" Verification and Error Checking

Support functions could be added to provide timely and interactive responses during live simulation exercises. The live exercise environment at the SMART conference stressed the need for real-time feedback and results checking. Table 7 identifies enhancements to support live exercise feedback and error checking.

Table 7. Live Exercise Support Functions

Function	Explanation
Display initial conditions	Display VL parameters that were part of the calculation of the lethality results.
Display damage source reference	Display information that identifies the reference that points to the damage source (the look-up table reference or other information that describes where the lethality data can be found) (i.e., display where the look-up table was found for a particular threat and munition).
Display the damage source	View the look-up table (or other vulnerability source data) (i.e., browse/view the data as opposed to where the data originated).
Entity Alert	Warns operator of a newly discovered entity for which there is no known vulnerability reference (i.e., no damage look-up table).
Munition Alert	Warns operator of a newly discovered entity for which there is no known vulnerability reference (i.e., "no damage look-up table was found").

Currently, almost all these enhancements are accessible by some means. That is, lethality server APIs already exist that include "debugging" print statements that can be "turned on" to examine how the server derived some result in as excruciating detail as is deemed necessary. However, this type of analysis is more suited for the controlled laboratory environment. To make some of these features readily available for immediate turn-around live analysis, these APIs need to be applied within the server, and their output must be presented in a usable manner. In some cases, new APIs will have to be added or old ones modified. Once applied within the server, the server and client interface libraries need to be modified to query and return the results of these new services. Finally, a GUI (such as the Java® client application) has to be modified or created to implement these functions and present the results in an intuitive and user-friendly fashion.

9. Summary

The RDEC Federation is a project that is bringing together many Army simulation assets for the purpose evaluating future concepts on a distributed virtual battlefield. It is a logical continuation and extension of the DoD's M&S distributed simulation research and capabilities.

The lethality server proved reliable in a live exercise with heavy data flows received; however, one of its primary design features (providing lethality results for all exercise participants) was not fully verified.

The SMART Conference provided the opportunity to conduct live trials on various additions to the server suite (such as the Java® client) and provide insights for improvements. Most of these improvements support the server's primary role of providing accurate and timely vulnerability results. Other improvements support the live exercise and distributed environment to a higher degree.

References

1. Sauerborn, G.C., K.G. Smith, A.W. Scramlin, R.R. Shankle, R.W. Gauss, W. Zhou, T.R. Perkins, P.E. Corcoran, J.A. Weller, R.W. Marvel, and J.P. Schimminger, "Project Focus: A Study of Virtual Proving Ground Software Architecture Requirements," ARL-TR-1429, U.S. Army Research Laboratory, Aberdeen Proving Ground, MD, September 1997.
2. Dunn, E., and G. Rumford, "TENA: A Domain-Specific Architecture for Live Participant," Ed Dunn and George Rumford, Paper: 00S-SIW-107, Simulation Interoperability Workshop Papers, The Simulation Interoperability Standards Organization (SISO), March 2000.
3. Pope, A., and R. Schaffer, "The SIMNET Network and Protocols," Report No. 7627, BBN Systems and Technologies Corporation, June 1991.
4. Institute for Electrical and Electronic Engineers, "Standard for Distributed Interactive Simulation - Application Protocols," DIS-4 Version 2.0 4th draft, (superseded by IEEE 1278.1), Institute for Simulation and Training, Orlando, FL, 4 February 1994.
5. Institute for Electrical and Electronic Engineers, "IEEE Standard for Distributed Interactive Simulation," IEEE 1278.[1235], 1995, 1996, 1997, 1998.
6. U.S. Department of Defense, "High-Level Architecture Rules Version 1.3," HLA-1, 5 February 1998 (20 April 1998 document release).
7. Sauerborn, G.C., "ARL Distributed Interactive Simulation (DIS) Lethality Communications Server, Volume I: Overview," ARL-TR-1775, p. 1, U.S. Army Research Laboratory, Aberdeen Proving Ground, MD, February 1999.
8. Deitz, P.H., and M.W. Starks, "The Generation, Use, and Misuse of 'PKS' in Vulnerability/Lethality Analysis," ARL-TR-1640, U.S. Army Research Laboratory, Aberdeen Proving Ground, MD, 1997.
9. Sauerborn, G.C., "ARL Distributed Interactive Simulation (DIS) Lethality Communications Server, Volume II: User and Programmer's Manual," U.S. Army Research Laboratory, Aberdeen Proving Ground, MD, February 1999.
10. Smith, K., "Distributed Interactive Simulation (DIS) Network Manger," ARL-TR-780, U.S. Army Research Laboratory, Aberdeen Proving Ground, MD, December 1994.

INTENTIONALLY LEFT BLANK

APPENDIX A
PHOTOS OF SIMULATION TESTING

INTENTIONALLY LEFT BLANK

PHOTOS OF SIMULATION TESTING

For historical purpose, the author gratefully acknowledges Oanh Tran of the Simulation, Training, and Instrumentation Command (STRICOM) for these and other excellent photos.

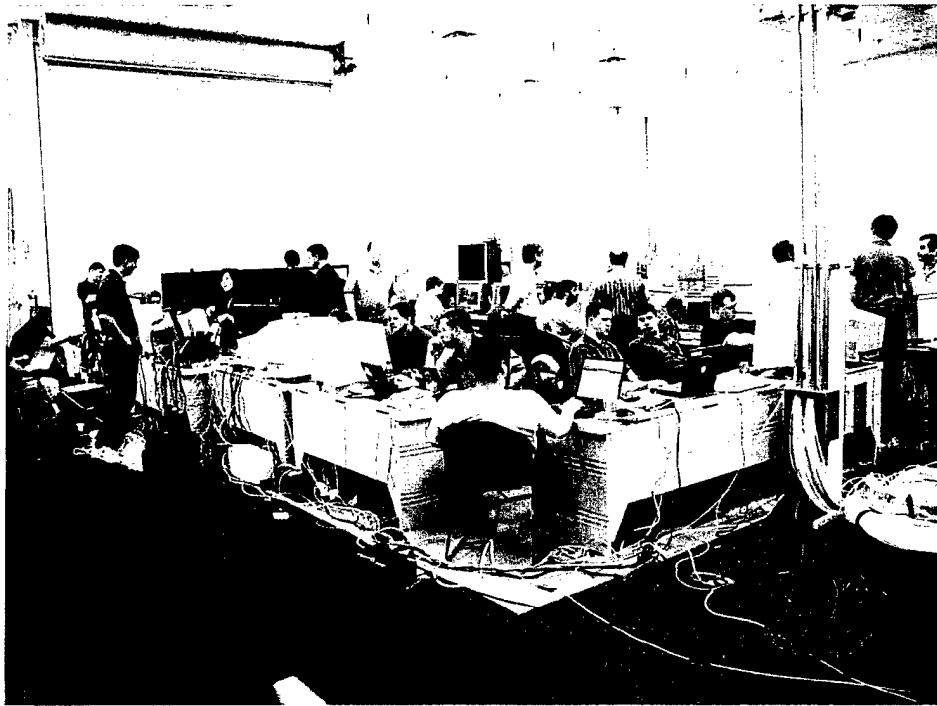


Figure A-1. RDEC Federation Integration Testing at STRICOM One Week Before the SMART Conference.



Figure A-2. On-lookers Observe a Live Exercise During the SMART Conference.

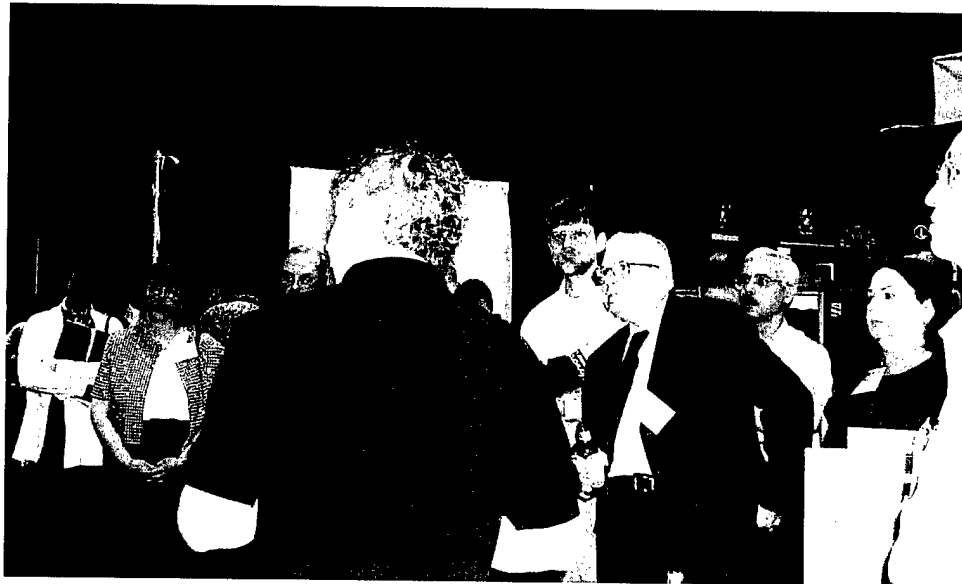


Figure A-3. During the Conference, Mr. Micheal Kelly (Night Vision Laboratory) Briefs Mr. Hollis Deputy Under Secretary of the Army (Operations Research), Dr. Bucher, AMCOM, Mr. Pei, CECOM, and Others.



Figure A-4. Selected Panoramic View 1 of the RDEC Federation During Conference Exercises.



Figure A-5. Selected Panoramic View 2 of the RDEC Federation During Conference Exercises.



Figure A-6. Selected Panoramic View 3 of the RDEC Federation During Conference Exercises.



Figure A-7. ARL-VPG Participants Left to Right: Gary Moss, Geoff Sauerborn, Mark Thomas.

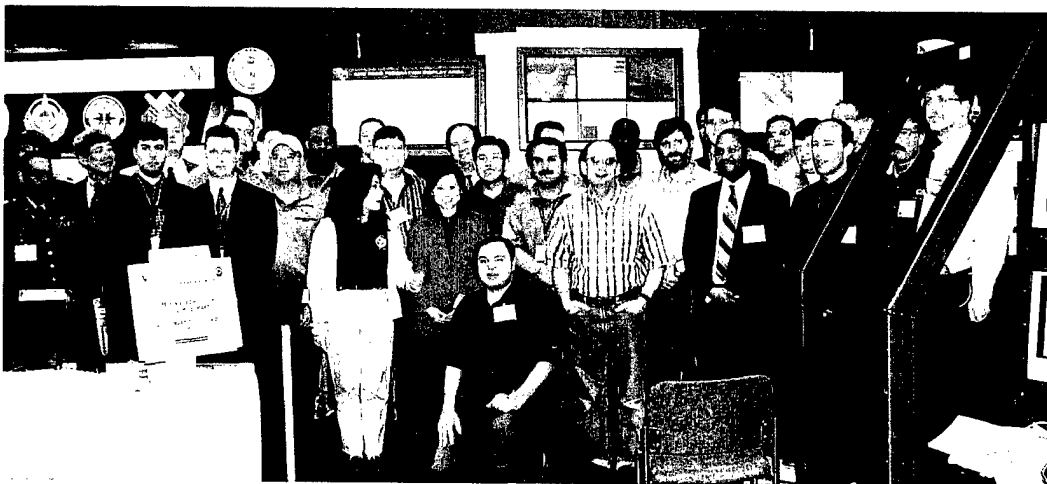


Figure A-8. RDEC Federation Participants Group Photo.

NO. OF
COPIES ORGANIZATION

1 ADMINISTRATOR
DEFENSE TECHNICAL INFO CTR
ATTN DTIC OCA
8725 JOHN J KINGMAN RD STE 0944
FT BELVOIR VA 22060-6218

1 DIRECTOR
US ARMY RSCH LABORATORY
ATTN AMSRL CI AI R REC MGMT
2800 POWDER MILL RD
ADELPHI MD 20783-1197

1 DIRECTOR
US ARMY RSCH LABORATORY
ATTN AMSRL CI LL TECH LIB
2800 POWDER MILL RD
ADELPHI MD 20783-1197

1 DIRECTOR
US ARMY RSCH LABORATORY
ATTN AMSRL D D SMITH
2800 POWDER MILL RD
ADELPHI MD 20783-1197

1 DOD JOINT CHIEFS OF STAFF
ATTN J39 CAPABILITIES DIV
CAPT J M BROWNELL
THE PENTAGON RM 2C865
WASHINGTON DC 20301

1 OFC OF THE DIR RSCH AND ENGRG
ATTN R MENZ
PENTAGON RM 3E1089
WASHINGTON DC 20301-3080

2 OFC OF THE SECY OF DEFNS
ATTN ODDRE (R&AT) G SINGLEY
ODDRE (R&AT) S GONTAREK
THE PENTAGON
WASHINGTON DC 20301-3080

1 AMCOM MRDEC
ATTN AMSMI RD W C MCCORKLE
REDSTONE ARSENAL AL
35898-5240

1 CECOM
SP & TERRESTRIAL COM DIV
ATTN AMSEL RD ST MC M
H SOICHER
FT MONMOUTH NJ 07703-5203

NO. OF
COPIES ORGANIZATION

1 US ARMY INFO SYS ENGRG CMD
ATTN ASQB OTD F JENIA
FT HUACHUCA AZ 85613-5300

1 US ARMY NATICK RDEC
ACTING TECHNICAL DIR
ATTN SSCNC T P BRANDLER
NATICK MA 01760-5002

1 US ARMY RSCH OFC
4300 S MIAMI BLVD
RSCH TRIANGLE PK NC 27709

1 US ARMY STRICOM
ATTN J STAHL
12350 RESEARCH PKWY
ORLANDO FL 32826-3726

1 US ARMY TANK-AUTOMOTIVE &
ARMAMENTS CMD
ATTN AMSTA AR TD M FISETTE
BLDG 1
PICATINNY ARSENAL NJ 07806-5000

1 US ARMY TANK-AUTOMOTIVE CMD
RD&E CTR
ATTN AMSTA TA J CHAPIN
WARREN MI 48397-5000

1 US ARMY TRADOC
BATTLE LAB INTEG & TECH DIR
ATTN ATCD B J A KLEVECZ
FT MONROE VA 23651-5850

1 NAV SURFACE WARFARE CTR
ATTN CODE B07 J PENNELLA
17320 DAHLGREN RD
BLDG 1470 RM 1101
DAHLGREN VA 22448-5100

1 DARPA
3701 N FAIRFAX DR
ARLINGTON VA 22203-1714

1 HICKS & ASSOCIATES, INC.
ATTN G SINGLEY III
1710 GOODRICH DR STE 1300
MCLEAN VA 22102

NO. OF COPIES	ORGANIZATION
1	SPECIAL ASST TO THE WING CDR 50SW/CCX CAPT P H BERNSTEIN 300 O'MALLEY AVE STE 20 FALCON AFB CO 80912-3020
1	HQ AFWA/DNX 106 PEACEKEEPER DR STE 2N3 OFFUTT AFB NE 68113-4039
1	APPLIED RSCH ASSOCIATES INC ATTN ROBERT SHANKLE 219 W BEL AIR AVE STE 5 ABERDEEN MD 21001
1	CDR US ARMY AVIATION RDEC CHIEF CREW ST R7D MS 243-4 (DR N BUCHER) AMES RSCH CTR MOFFETT FIELD CA 94035
1	ITT INDUSTRIES ATTN CHARLES WOODHOUSE 2560 HUNTINGTON AVE ALEXANDRIA VA 22303
1	ITT INDUSTRIES ATTN MICHAEL O'CONNOR 600 BLVD SOUTH STE 208 HUNTSVILLE AL 35802
1	RAYTHEON SYSTEMS COMPANY ATTN JOHN D POWERS 6620 CHASE OAKS BLVD MS 8518 PLANO TX 75023
1	OPTOMETRICS INC ATTN FREDERICK G SMITH 3115 PROFESSIONAL DR ANN ARBOR MI 48104-5131
1	DIR US ARL ATTN AMSRL SL EP G MAREZ WSMR NM 88002
5	DIR US ARMY TRAC ATTN ATRC WE L SOUTHARD ATRAC WEC J AGUILAR C DENNY D DURDA P SHUGART WSMR NM 88002

NO. OF COPIES	ORGANIZATION
3	CDR TARDEC ATTN AMSTA TR D M/S 207 FSCS R HALLE G SIMON WARREN MI 48397-5000
3	CDR ARDEC ATTN AMSTA AR FSS J CHU D MILLER B DAVIS PICATINNY ARSENAL NJ 07806-5000
1	DEF THREAT REDUCTION AGENCY ATTN SWE W ZIMMERS 6801 TELEGRAPH RD ALEXANDRIA VA 22310
2	JOINT VIRTUAL BATTLE SPACE ATTN MAJ R SCHWARZ J GARCIA 10401 TOTTEN RD BLDG 399 STE 325 FT BELVOIR VA 22060-5823
1	U.S. SBCCOM NATICK SOLDIER CENTER ATTN AMSSB RSS MA (N) D TUCKER KANSAS STREET NATICK MA 01760-5020
2	HQ OPERATIONAL TEST CTR ATTN CSTE OTC MA S CSTE OTC MA S J HAMILL BLDG 91012 FT HOOD TX 76544-5068
1	SANDIA NATL LABORATORIES ATTN M J MCDONALD PO BOX 5800 MS-1004 ALBUQUERQUE NM 87185-1004 <u>ABERDEEN PROVING GROUND</u>
2	DIRECTOR US ARMY RSCH LABORATORY ATTN AMSRL CI LP (TECH LIB) BLDG 305 APG AA
8	DIR AMSAA ATTN P DEITZ M BORROUGHS B BRADLEY D HODGE DON JOHNSON A WONG BLDG 392

NO. OF
COPIES ORGANIZATION

1 DIRECTOR
US ARMY RSCH LABORATORY
ATTN AMSRL WM J SMITH
BLDG 4600

1 DIRECTOR
US ARMY RSCH LABORATORY
ATTN AMSRL WM B A HORST
BLDG 4600

1 US ARMY RSCH LABORATORY
ATTN AMSRL WM BE R SANDMEYER
BLDG 328

5 US ARMY RSCH LABORATORY
ATTN AMSRL WM BE L BUTLER
J ANDERSON R BOWERS
C KENNEDY P TANENBAUM
BLDG 238

75 DIRECTOR
US ARMY RSCH LABORATORY
ATTN AMSRL WM BF J LACETERA
G SAUERBORN (74) CYS
BLDG 309

2 DIRECTOR
US ARMY RSCH LABORATORY
ATTN AMSRL CI CT G MOSS
M THOMAS
BLDG 321

INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY (Leave blank)		2. REPORT DATE December 2001		3. REPORT TYPE AND DATES COVERED Final	
4. TITLE AND SUBTITLE Modifications of the Lethality Server for Initial RDEC Federation Integration				5. FUNDING NUMBERS PR: AH 43 and AH80	
6. AUTHOR(S) Sauerborn, G.C. (ARL)					
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) U.S. Army Research Laboratory Weapons & Materials Research Directorate Aberdeen Proving Ground, MD 21005-5066				8. PERFORMING ORGANIZATION REPORT NUMBER	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) U.S. Army Research Laboratory Weapons & Materials Research Directorate Aberdeen Proving Ground, MD 21005-5066				10. SPONSORING/MONITORING AGENCY REPORT NUMBER ARL-MR-522	
11. SUPPLEMENTARY NOTES					
12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.				12b. DISTRIBUTION CODE	
13. ABSTRACT (Maximum 200 words) This report summarizes recent changes in the U.S. Army Research Laboratory distributed interactive simulation lethality communications server (the lethality server) and its integration into the Research, Development, and Engineering Center Federation.					
14. SUBJECT TERMS distributed interactive simulation (DIS) lethality vulnerability high level architecture (HLA) RDEC				15. NUMBER OF PAGES 40	
				16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified		20. LIMITATION OF ABSTRACT	